IN THE SPECIFICATION:

Please replace the paragraph that starts at the bottom of page 8 and ends at the top of page 9 with the following paragraph:

Computer 30 operates under the control of an operating system 40, and executes various computer software applications, components, programs, objects, etc., such as an executable program 42, and perhaps a repository 44, a constructor 50, a matchmaker 60 and an announcer 70, as will be discussed. The constructor 50, the matchmaker 60, and announcer 70 are [[is]] resident in memory 32 for the purpose of evaluating one or more listings of computer programs, e.g., executable program 42. These and other various applications components, programs, objects, etc., may also execute on one or more processors in another computer coupled to computer 30 via a network 38, e.g., in a distributed or client-server computing environment whereby the processing required to implement the functions of repository 44, constructor 50, matchmaker 60, and announcer 70 may be allocated to multiple computers over a network.

Please replace the paragraph that begins with the words "Figure 4 further elucidates" on page 11 with the following paragraph:

Figure 4 further elucidates the process outlined in step 340 of FIG. 3 of obtaining the construct list. From start at block 410, the constructor extracts tokens from the changed construct/portion of the code as in step 420. In step 430, the next construct in the construct list is parsed. The method determines if the construct is a workable size in the blocks 435 and 440. In block 435, the constructor checks if the construct is too small and the size of the construct is below a minimal threshold, then it is not considered. Actually, considering the size of the construct for purposes of comparing with other constructs is somewhat of an art. If the construct is too small, such as a simple ***if*** ~~$\boldsymbol{a}$-^{1}$\boldsymbol{0}$~~ $\underline{\boldsymbol{a}^{10}\ \ }$, then the construct list would be too large, there would be too many matches, and system of matching herein would not be useful. On the other hand, if the construct is too large, e.g., perhaps hundreds of lines of code including several other constructs, then the constructor might not be able to identify any similar or derived code. So, the constructor uses a configurable threshold value to set a minimal size for the matching code segments. For instance, some embodiments may set this minimal size to be fifteen tokens. To complete the inquiry, in step 440, the constructor determines if the construct ~~to~~ is too large. If so, then the larger construct is partitioned into its smaller constructs as in block 450. If, however, the construct is of a reasonable size, as in step 460, the construct is added to the list of constructs.

Please replace the paragraph that starts with "With reference to Figure 6" on page 12 with the following paragraph:

With reference to FIG. 6, once the list of constructs has been identified, the matchmaker reviews those constructs similar to or related to or derived from the changed construct to determine if any of the constructs **match** with the construct that has changed. **Match** is a relative term and can be varied according to the tokens within the construct as will be further discussed. From start at 610, each source file is further evaluated, as in steps 612, 686 and 688 ~~step 620~~. For example, after obtaining the first changed source unit or construct in a program (step 612) and scrutinizing it (see steps 630 – 682) then the next changed source unit or construct in the program will be obtained and scrutinized (see steps 686, 688 and 614). If there is not a next changed source unit or construct in the program, then the process exits (steps 686 and 690). The matchmaker looks for sections of the same or similar code within the various source files, or even within a single source file. The source files examined to find these sections may be all the parts in a particular component, all the parts in a particular software product, or all the parts that exist in a particular source repository. For each source unit, its construct list is retrieved, as in block 614 ~~630~~, and for each construct in the list, steps 630, 682 and 684 ~~step 640~~, the matchmaker will determine as in step 650 if there is a match as in step 660, i.e., if the construct under evaluation is so similar to the code that was changed. It has been noted by the inventors that if there a high degree of matching, if there is minimal data flow into and out the construct, and if there are enough of these matching constructs, then a developer may wish to consider changing the construct to a method, such as in object oriented programming. To practice these teachings, then, at step 665 there is an inquiry if the match is high enough to consider creating a procedure or a method. If yes, then at step 670, the constructor will examine the data flow and determine if the data flow in/out of the construct is acceptable for a method or procedure. If so, then at step 675, the user is notified that she/he might increase the efficiency of the code if they wrote a method or procedure having the same function/code as the construct under evaluation. Then, the owner is also notified in step 680 that the matching derivative or related code has changed. The matchmaker may determine a match exists using at least two different techniques.